EGI-Engage

Data flow handler and basic R tools to integrate and process data from Ecological Observatories in EGI

D6.6

Date	20 February 2016
Activity	WP6
Lead Partner	CSIC
Document Status	DRAFT
Document Link	https://documents.egi.eu/document/2666

Abstract

The LifeWatch EGI Competence Center is oriented to capture and address the requirements of Biodiversity and Ecosystems research communities. In this deliverable we report on the experience handling data flows from two Ecological Observatories, a marine vessel instrumented with different sensors, and an instrumented profiling platform in a water reservoir. Also the use of the tools that can be used to integrate and process those data is analyzed, in particular the experience on the deployment and use of R-based tools. The different services installed are described, and also first feedback from users is provided.

The EGI-Engage project is co-funded by the European Union (EU) Horizon 2020 program under Grant number 654142

COPYRIGHT NOTICE

DELIVERY SLIP

	Name	***Partner/Activity***	***Date***
From:	Jesus Marco	CSIC/SA2	20 Feb 2016
Moderated by:			
Reviewed by			
Approved by:	AMB and PMB		

DOCUMENT LOG

Issue	***Date***	***Comment***	***Author/Partner***
v.1	04/05/2015	First draft	F.Aguilar/CSIC
v.2	22/01/2016	Formatted	E.Fernández/EGI
v.3	02/02/2016	R tools description, Previous experiences.	F.Aguilar, A.Palacios /CSIC
v.4	04/02/2016	Introduction, Context	J.Marco/CSIC
v.5	12/02/2016	HCMR Input	A.Oulas, E. Panteri /HCMR
v.6	16/02/2016	VLIZ Input	F.Hernández/VLIZ
v.7	17/02/2016	Integrated version	F.Aguilar/CSIC
v.8	20/02/2016	Revised version, ready for internal review	J.Marco/CSIC
v.9	26/02/2016	Version including comments(GS, AB, FB,AJPL)	J.Marco/CSIC
v.10	27/02/2016	Version ready for delivery	J.Marco/CSIC

TERMINOLOGY

A complete project glossary is provided at the following page: http://www.egi.eu/about/glossary/

Contents

Executive summary

The goal of the LifeWatch EGI Competence Center (LW-EGI-CC) is to capture and address the requirements of Biodiversity and Ecosystems research communities. To achieve this goal, the LW-EGI-CC has been working on the definition, development and deployment of the Cloud and GPGPU based e-Infrastructure services that are required to support data management, data processing and modelling for Ecological Observatories.

In this deliverable we report on the experience supporting the data flow from different Ecological Observatories into EGI e-infrastructure, and then on the tools that can be used to integrate and process those data, and in particular on the deployment and use of R.

Two different ecological observatories are already providing data into EGI FedCloud resources via LW-EGI-CC:

-Flanders Marine Institute (VLIZ) has installed a number of biosensors on board of the Research Vessel Simon Stevin, as part of the Flanders Marine LifeWatch Observatory, providing a data flow that will reach about 50Tb of data per year, mainly video and images, collected by the vessel in quasi real time and requiring a substantial computational power, to incorporate a framework based in R for the final researcher.

-IFCA and a Spanish SME (Ecohydros SL) have been operating for the last five years an advanced monitoring platform in a water reservoir to detect cyano-algae blooms, that is providing a continuous data flow and requires also the integration of external data into EGI FedCloud, used by the SME researchers to contrast the modelling tools. R is used systematically to provide to the online monitoring with the computation of relevant quantities like the vertical temperature profile parameters evolution (epilimnion/hypoliminion parameters among many others).

A detailed analysis of the possibilities to implement and deploy services oriented to support the use of R is also presented, starting from the previous experience in the Grid framework (processing data from the LTER Observatory of Sierra Nevada in Spain), describing the implementation in HPC systems, in clusters in other LifeWatch centers (HCMR, VLIZ, IFCA), and also starting the discussion on how to compare the performance in order to improve it combining the experience and different approaches of the different teams in the LW-EGI-CC.

As a demo-oriented deliverable, references to access these services are provided, and also a brief report based on the users' experience that includes some proposals for evolution and improvement.

Also a proposal for the architecture to implement R as a service in the EGI FedCloud infrastructure is formally presented.

1 Introduction

Ecological Observatories are one of the key data providers for LifeWatch, and a clear objective of the LifeWatch EGI Competence Center[1] (LW-EGI-CC) is to integrate the tools required to support data management, data processing and modelling for Ecological Observatories in the framework provided by EGI.eu. To achieve this objective, several Case Studies directly related to on-going LifeWatch initiatives that require the manipulation of data streams from different Ecological Observatories have been considered and analysed.

Ecological Observatories data processes require today more and more "Big Data" techniques, from support to real-time data streams to handling the post-processing of large volumes of diverse data from multiple disciplines: meteorology, geophysics, hydrology, chemistry, social and of course life sciences (biology, ecology, -omics). Moreover, in the last years new and powerful software packages are starting to allow also the simulation of these complex multidisciplinary systems.

The LW-EGI-CC promotes the use of the EGI FedCloud as a common e-infrastructure to the different related initiatives, as it has been adopted as a basic platform for distributed e-infrastructure by the LifeWatch initiative, offering both the (substantial) resources required and also the possibility to test and install Cloud solutions at SaaS, PaaS and IaaS level. The LifeWatch Virtual Organization (LW-VO) is used to support this work, and as explained later, different roles are supported using different authentication and authorization mechanisms. Different FedCloud sites are supporting now the LW-VO, in particular the site at IFCA (Instituto de Fisica de Cantabria, CSIC-University of Cantabria), and the new site at EBD (Estacion Biologica de Doñana, CSIC) offer relevant computing and storage cloud enabled resources, at the different IaaS (Infrastructure as a Service), PaaS (Platforma as a Service) and SaaS (**Solution**/Software as a Service) levels.

Within this LW-EGI-CC framework, and in order to analyze how to offer an adequate support to the researchers' requirements exploiting this FedCloud framework, we have considered two complete Case Studies that are introduced in what follows below.

The first one corresponds to the monitoring of a Water Reservoir, developed through a joint effort of an Spanish environmental consultancy SME (Ecohydros SL) and IFCA under the umbrella of LifeWatch Spain. Eutrophication, resulting in algae bloom, is an increasing serious problem in many water reservoirs in Europe and in the whole world due to the increase of anthropogenic pressure (human activities, including also farming) and climate change (warmer summers favour algae bloom). The prediction of eutrophication and of the development of algae bloom requires modelling the water reservoir from the hydrological perspective, predicting in detail the temperature profile of the water and its composition, and also the modelling of all processes related to algae growth from the biological point of view. The validation of this complex model requires historical measurements from a complex in-situ instrumentation.

[1] See https://wiki.egi.eu/wiki/CC-LifeWatch_Community

The current setup includes a central platform that is installed in the middle of the water reservoir, and instrumented with meteorological sensors (wind, temperature, solar radiation, rain, etc), and water quality sensors (conductivity, temperature, dissolved oxygen, turbidity, pH, etc.). The water quality sensor probe is placed in a cage connected to a wincher system allowing vertical profiling (range 1-30 m. in depth) that is critical to monitor the evolution of the water stratification, clearly reflected in the thermocline curves. More complex instrumentation, including radiometers, spectrometers and absorbance sensors are also included to monitor the abundance of green and blue-green algae, through the correlation with the luminescence of chlorophyll and phycocyanin. All data gathered is stored in databases and can be accessed with different tools, including a visualization dashboard with an option enabling users to download data easily. The original "raw" data stored is then "processed" and some derived parameters are calculated using different methods, including the use of R-based scripts. Currently more than 5 years of data have been collected and analyzed[2]. Another key ingredient for modeling this water reservoir is the simulation model and the corresponding software suite. Currently the solution used is the open source suite Delft3D[3] that includes a module providing the simulation of the hydrodynamics of the water reservoir (FLOW) and another module for the simulation of the water quality (DELWAQ).

A different Ecological Observatory considered is related to an ongoing experience by LifeWatch Belgium on marine biodiversity research. This field of research is very dependent on specific data types: species descriptions and identifications, their behavior, occurrences, presence/absence, biomass, abundance and many others similar. For a long time, the collection of this type of data has been mainly a manual process: sampling, sample preparation, identifying species, counting, weighing, typing the data in spreadsheets or databases etc. The use of biosensors and sensor networks for in-situ observation seems to be one of the most promising approaches as this method eliminates the need for taking physical sampling and avoids labor intensive sample preparation processes; moreover, the dataflow can be automated and requires less workload from the scientists. Following this promising approach, Flanders Marine Institute (VLIZ) has promoted the installation of a number of biosensors on board of the Research Vessel Simon Stevin, as part of the Flanders Marine LifeWatch Observatory. This project has a series of needs that require the use of a powerful e-infrastructure able to handle a "Big Data"-like problem: about 50Tb of data per year, mainly video and images, will be collected by the vessel in quasi real time, and their analysis requires a substantial computational power, that will be provided by the EGI-FedCloud e-infrastructure. This project also needs to incorporate an analysis framework for the final researcher that includes R-tools.

To summarize, both Case Studies offer interesting examples of the application of new "on-line" instrumentation to ecological observatories, requiring the integration of data into the e-infrastructure, and the corresponding post-processing, and in particular the use of R-based analysis tools by the researchers.

[2] For a complete overview see: A. Monteoliva, PhD Thesis, Feb 2016 (in press)
[3] DELFT3D: see http://oss.deltares.nl/web/delft3d

2 Handling Data Flows

In this section we describe how the "Data Flows" are managed in both case studies. This is an on-going work, where very substantial progress has been achieved solving different practical problems, like handling the very large collections of very small image files, or establishing a reliable connection to a remote observation site.

Marine Data Stream

Figure 1 describes the global flow of data in the Case Study corresponding to the marine ecological observatory. Data collected in the vessel is transmitted to VLIZ research center offices, ingested using the corresponding standards, and transmitted to the FedCloud e-infrastructure for replication, and it is made available for further processing.

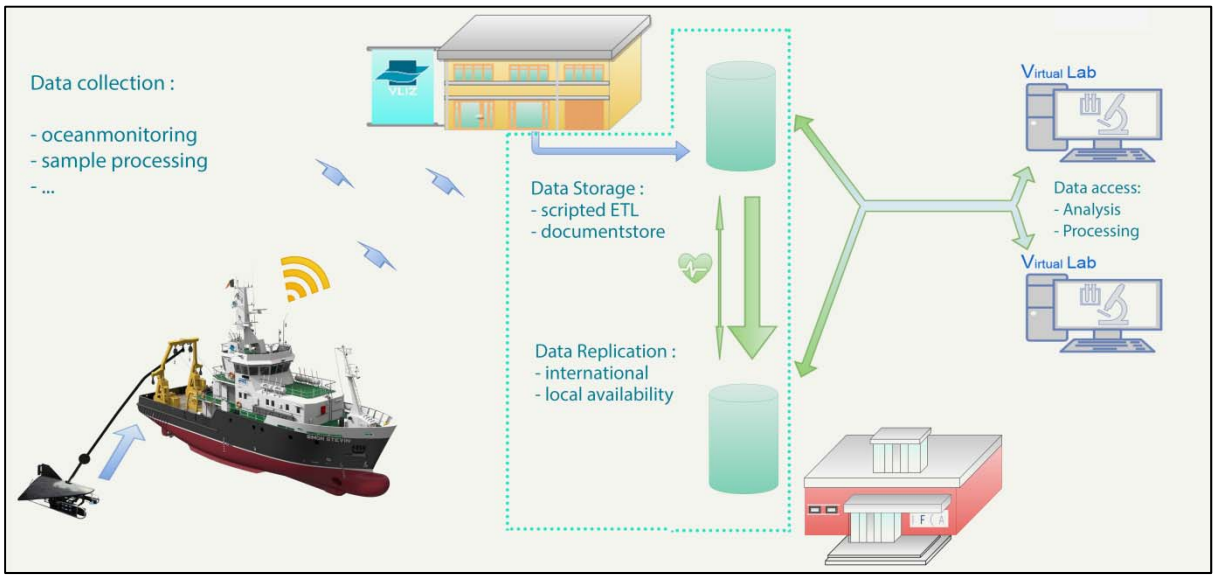

Figure 1 – Data Flow in the Case Study of the marine observatory managed by VLIZ center

The substantial progress in the implementation of this data flow, thanks to different solutions, is described in what follows:

Data synchronization

The synchronization of data from the marine station (VLIZ), and the research vessel to the EGI servers (IFCA) are partly operational. Specifically:

-From Research Vessel to VLIZ archive disks: this is operational for all installed sensors.

-From VLIZ archive disks to MongoDB: this is operational for ZooScan data and VPR data, developments for FlowCytometer data have started.

-From VLIZ MongoDB server to EGI MongoDB server : this is operational for ZooScan and VPR data.

Data accessibility

The MongoDB databases in VLIZ and IFCA are accessible from the Rshiny/ Rstudio based virtual lab running at VLIZ: the LifeWatch data explorer. In a next phase the LifeWatch data explorer server will be duplicated to IFCA for increased processing power.

Demonstration website accessing server at VLIZ: http://rshiny.lifewatch.be/ZooScan%20data/

Access to files through MongoDB

The virtual labs should also have access to the individual files generated by the different biosensor instruments. The following diagram explains how this will be achieved. We need to reconstruct the files and folders organisation because we want to use existing analysis software, that requires file based access to the data. The first part of this mechanism (files to mongodb) is operational. The second part (the mongoDB to files API) will be dealt with in the next phase.

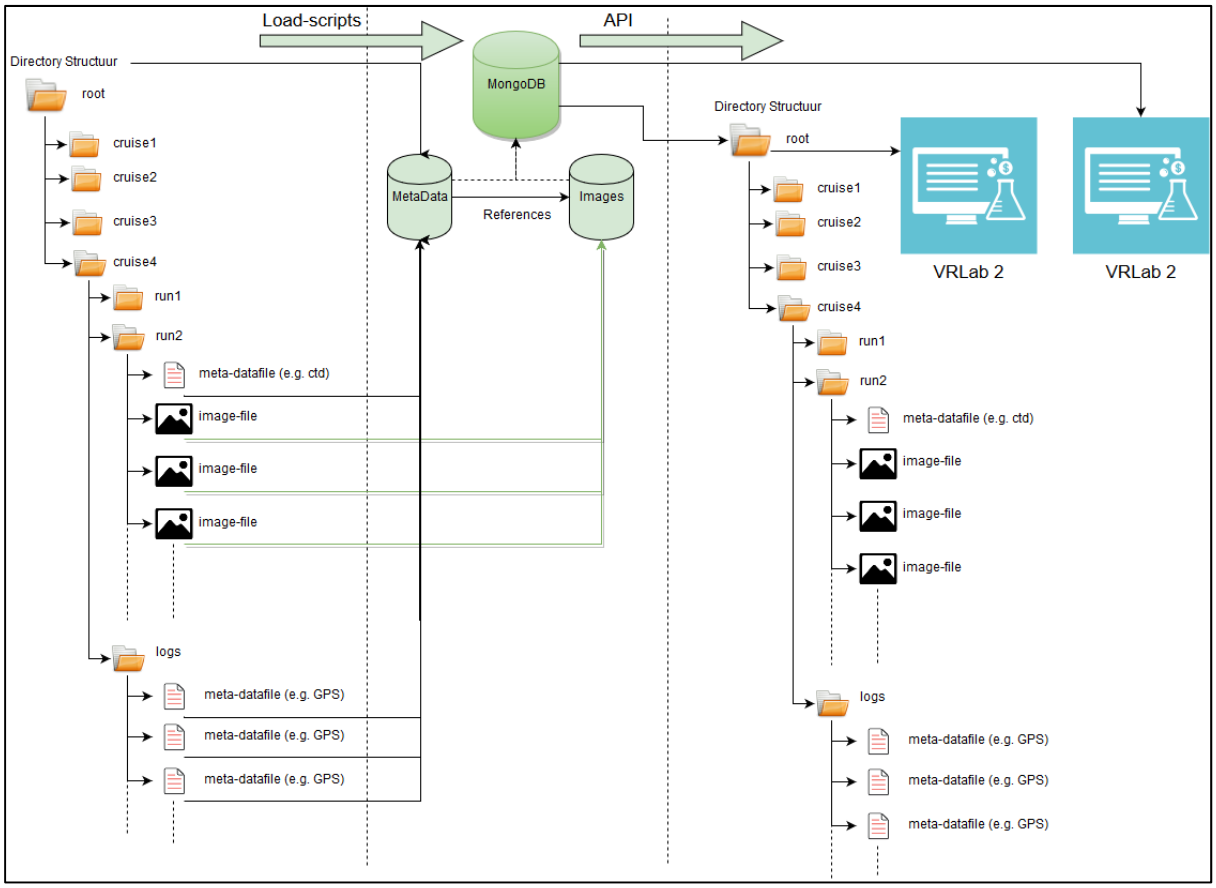

Figure 2 – MongoDB & Images Management

Access to data in SQL databases

This part is operational for station data, underway data, buoy data, bird tracking data and fish tagging data, and for access to the data from LifeWatch data explorer.

Demonstration here: http://rshiny.lifewatch.be/

Access from the IFCA virtual lab will be implemented in the next phase after we solve the data synchronization and security problems.

Access to data through Geoserver webservices

Prototype to access Geoserver data through WFS calls from the Rshiny/Rstudio virtual lab was built and feasibility demonstrated. Using Geoserver clusters could boost the speed of accessing data. This is ongoing work within the Geoserver working group in LW-EGI CC.

CdP Water Reservoir Data Flow

The scheme followed to collect data from the CdP (Cuerda del Pozo) Water Reservoir is described in Figure 3.

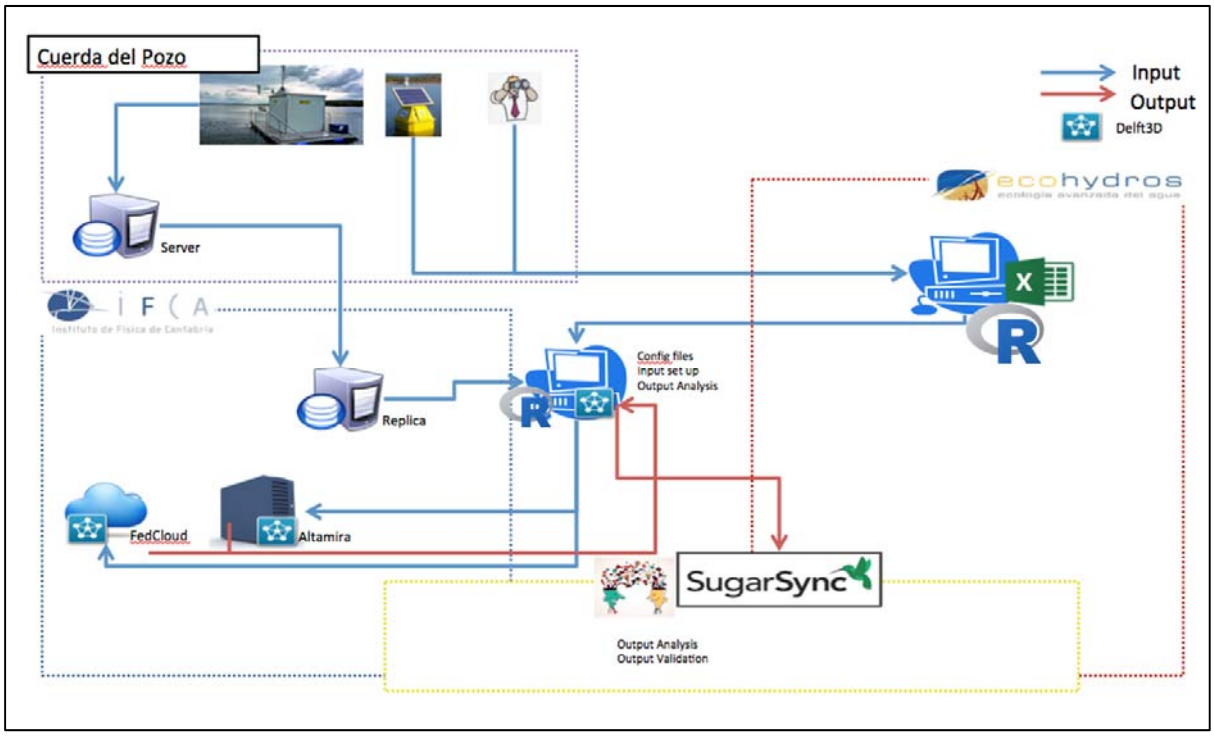

Figure 3 – CdP Data Flow Schema

Currently, the data flow has two main sources: "real-time" instrumentation (platform and buoys) and "external" information, that includes a wide range, from bathymetry maps to analytics on the composition of the different bed layers. All these different data is required to build a complete model for the water reservoir. The "real-time" data is directly stored as "RAW" in the databases, and then corrected and filtered to be published as "PROCESSED" via a web portal (http://doriiie02.ifca.es/) . The "external" information is stored as files and also in spreadsheets and it also has to be corrected, filtered and in some cases validated (internally or externally contrasted with other reference measurements).

These "processing" steps are in some cases implemented directly on the databases or spreadsheets, but as the data collections become larger (as an example, meteorological and solar radiation measurements have been collected each 10 minutes, along 5 years) they require the use of more powerful statistical tools like those based on R. In particular R scripts are executed directly on databases (using the RMySQL package) and also on text-based files.

It is quite important to understand that part of these data is required as input to the quite complex hydrodynamic and water quality models implemented in the Delft3D modeling suite, that in turns provided the expected abiotic (water temperature and other relevant physical parameters 3D profiles across the whole water reservoir) and biotic (algae concentration, and corresponding oxygen, N and P profiles) information. This modelling software produces large output files with these predicted parameters that can be exported in different formats, including ".csv", that are then analyzed using R tools. The final stage is the comparison of the real data with this output, to validate the model, and fine-tune the parameters required to make the corresponding predictions using the new inputs being collected (for example at the start of a potential eutrophication period for the water reservoir in August each summer). This is of direct interest for the main stakeholder in the project, the water authority managing the water reservoir (Confederacion Hidrografica del Duero, CHD).

3 Processing Data: R based analysis

Introducing R

R is a programming language and software environment[4] for statistical computing and graphics designed by Ross Ihaka and Robert Gentleman. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. R's popularity has increased substantially in recent years since its start in 1993.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

Many of R's standard functions are written in R itself, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks, C, C++, and Fortran code can be linked and called at run time. Advanced users can write C, C++, Java, .NET or Python code to manipulate R objects directly.

R is highly extensible through the use of user-submitted packages for specific functions or specific areas of study. R has stronger object-oriented -programming facilities than most statistical computing languages.

R provides a wide variety of statistical (linear and non-linear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques. It includes:

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hard copy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

R applications are used for theoretical computational statistics and the hard sciences such as astronomy, chemistry and genomic to practical applications in business, drug development, finance, health care, marketing, medicine and much more. Examples of applications are:

- Chemometrics and Computational Physics

[4] This section is directly adapted from https://www.r-project.org/about.html and it is included here for the interest of readers not familiar with R.

- Clinical Trial Design, Monitoring, and Analysis
- Computational Econometrics
- **Analysis of Ecological and Environmental Data**
- Design of Experiments & Analysis of Experimental Data
- Statistical Genetics
- Medical Image Analysis

R is enriched with a number of community packages that can be easily extended and deployed, enabling users to include more functionalities like parallelization, interaction with other languages, support to different data formats analysis, etc.

Support for R

In 2007, the company Revolution Analytics was founded to provide commercial support for Revolution R, its own distribution of R, which also includes other components developed by the company including Parallel R, the R Productivity Environment IDE, etc. In 2015, Microsoft Corporation completed the acquisition of Revolution Analytics.

In October 2011, Oracle announced the *Big Data Appliance*, which integrates R, Apache Hadoop, Oracle Linux, and a NoSQL database.

IBM offers support for in-Hadoop execution of R, and provides a programming model for massively parallel in-database analytics in R.

Other major commercial software systems supporting connections to or integration with R are JMP, MATLAB, Spotfire or Tableau.

A previous experience using R in the EGI.eu framework

R allows users to handle different data formats, included “geo” formats like NetCDF and others, and this possibility is widely exploited by the ecological and biodiversity community. Within the LifeWatch initiative and under previous related projects, R usage has been tested in different platforms both HTC (High Throughput Computing) using European Grid Initiative (EGI) resources and HPC (High Performance Computing) resources, using a top500 supercomputer (Altamira).

As a relevant example we would like here to comment on a pilot project aimed to prospect techniques in temporal series to classify vegetation cover in a given region automatically. The NDVI (Normalized Difference Vegetation Index) is an index that allows estimating quantity, quality and vegetation growing through remote sensing, in particular satellite images. The project used images provided by the MODIS sensor in NASA's TERRA satellite, where each pixel corresponds to an area of 250 x 250 m. The study covered an area extending from the French Pyrenees to the South of Spain (3 images in total).

Figure 4 – Satellite images processed in the example

After selecting the images of interest, pattern recognition techniques can be applied to analyze the vegetation cover as this information is useful to researchers to develop distribution models, anomalies, fire detection, etc. In this particular example, the analysis used a Geographical Information System (GIS), GRASS, and a library for reading and writing geospatial data, GDAL. Both solutions required specific packages for interacting using R: *rgdal* and *rgrass*. The processed MODIS images (HDF format) were stored in a PosgreSQL database that was used calculate specific snow and NDVI indicators. Finally the workflow used an ontology to describe new relationships among ecological concepts.

Combining both statistical and GIS tools is a very common and relevant need in biodiversity and ecological research communities, and as said, R is a complete and powerful solution that provides all needed tools to handle data, analyse it, process it and show the output in many different forms like charts, graphs or maps.

This complex and interesting analysis that explored the use of EGI resources for this work was published in 2015 in the International Journal of Applied Earth Observation and Geoinformation **37**, 142–151, A.J.Perez-Luque et al.: "An ontological system based on MODIS images to assess ecosystem functioning of Natura 2000 habitats: A case study for Quercus pyrenaica forests".

Analysis of R tools

In this section we explore different R tools and analyze the possibility of integrating them as services to be offered in the LW-EGI-CC framework, using FedCloud resources. This is not an exhaustive survey, rather we focus on a few that could be offered directly to the researchers, at different levels, to complete their data analysis work.

RStudio Server

RStudio[5] is an IDE (Integrated Development Environment) specially oriented to develop R programs and use the potential of this language for performing statistics and data analysis. RStudio is available in two different editions: one for the desktop environment and another one oriented to be used via a web interface: RStudio Server. We consider that the Server option is more interesting for the Fedcloud framework.

RStudio includes four different and customizable components that make it easier to work with data (see the figure below): a console where the user can insert the different commands, an editor to set up scripts directly, a data objects management box, and finally another box for different tasks, like showing graphical results. RStudio also includes different tools for debugging, workspace management, package installation, etc.

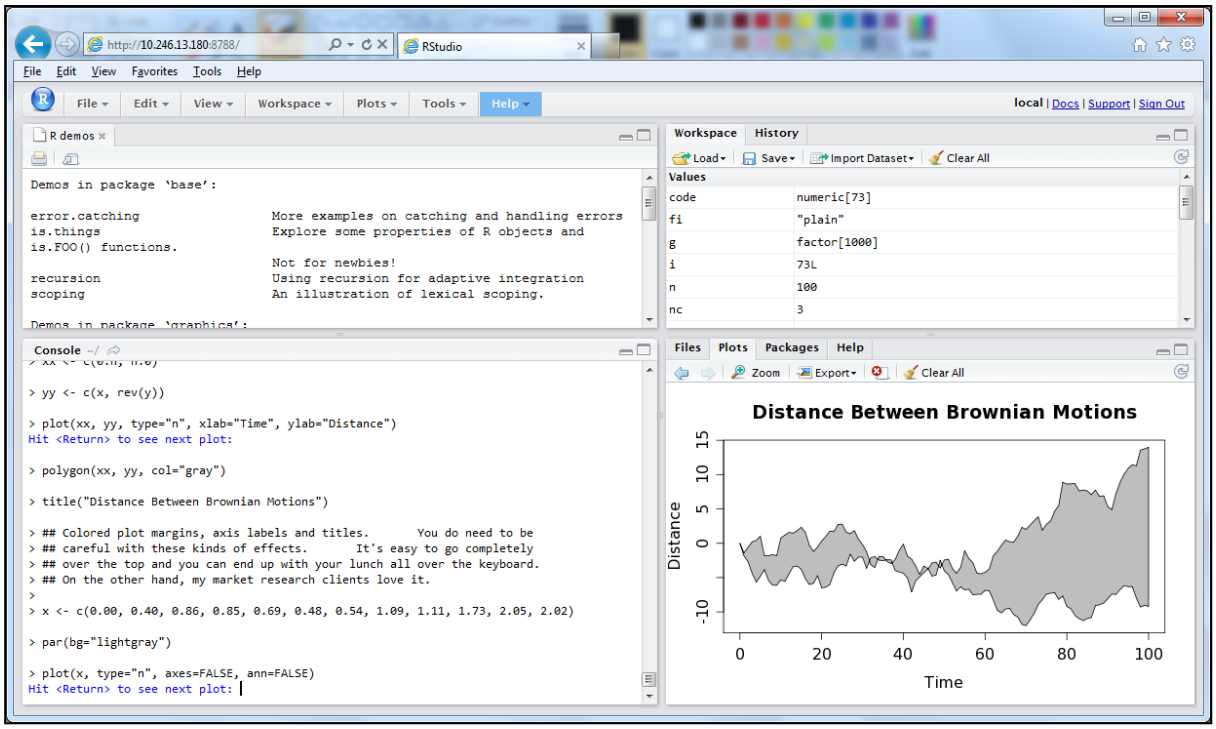

Figure 4 – RStudio IDE

[5] See https://www.rstudio.com/

RStudio server is available under two different versions (and licenses): Open Source and Enterprise. In the Open Source version, the main functionalities are included as well as the execution of R in local resources (local to the server). The Enterprise version includes additional interesting functionalities like load balancing across nodes, user and group management, metrics and monitoring, etc. However, some of these features can be found under different R packages like for example Rserve[6], which is able to connect to different nodes to process R scripts.

RShiny

RShiny[7] is a framework supporting the development of R-based applications through a graphical user interface. Oriented to the final user, Rshiny includes many relevant R features, including the access to different packages without the need to develop R code.

Figure 6 shows an example of the use of Rshiny with map oriented packages.

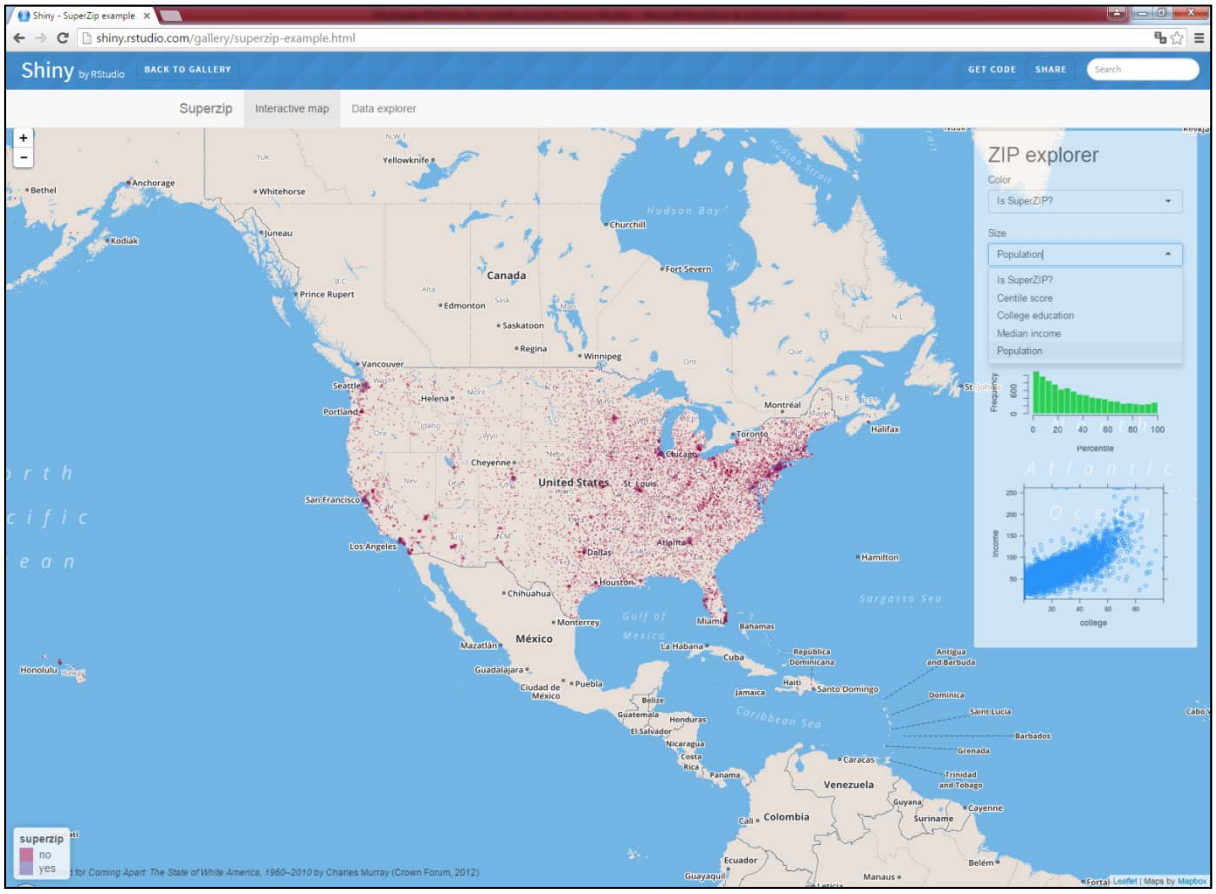

Figure 5 – Rshiny example

[6] See http://www.rforge.net/Rserve/
[7] See http://shiny.rstudio.com/

Rshiny is completely dynamic, and the user can interact with the different elements in the screen, such as drop-down lists, text boxes, spinners and even the script itself can be changed dynamically. The framework works over a web server and offers a web interface, that is accessed by the final users via a browser. A second example showing how a very simple but relevant data analysis can be implemented is displayed below.

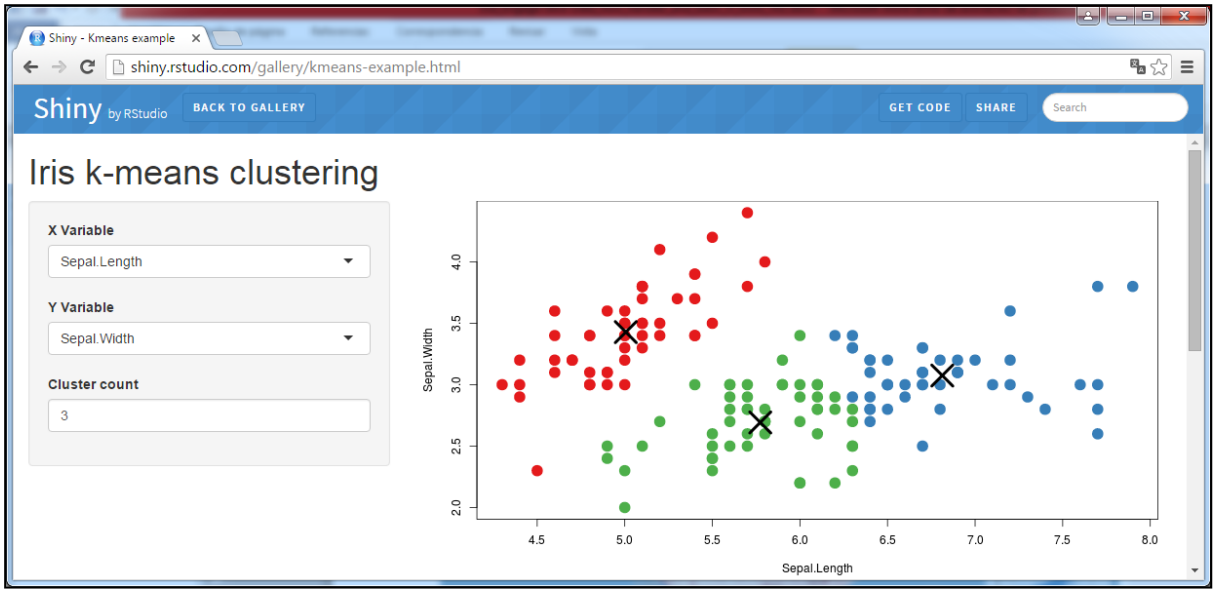

Figure 6 – Another example using RShiny: k-means clustering on bidimensional data

Jupyter Notebook

An interesting alternative to the RStudio framework, the Jupyter Notebook[8] is an interactive web application where the user can create and share scripts/programs written in many different languages, access and process data, and visualize the corresponding output.

Jupyter is based in the iPython Notebook[9], a web application developed by the Python community following the look and feel of Mathematica notebooks, integrating rich data representations and even figures with publication quality, in the web browser. Jupyter extends this approach to support many other languages, including R.

Jupyter installation requires activation of a kernel for each language/library to be used, the IRkernel in the case of R, and also the corresponding connection.

Figure 8 shows an screenshot of a Jupyter notebook being executed in a server using the R flavor.

[8] See http://jupyter.org/
[9] See http://ipython.org/notebook.html

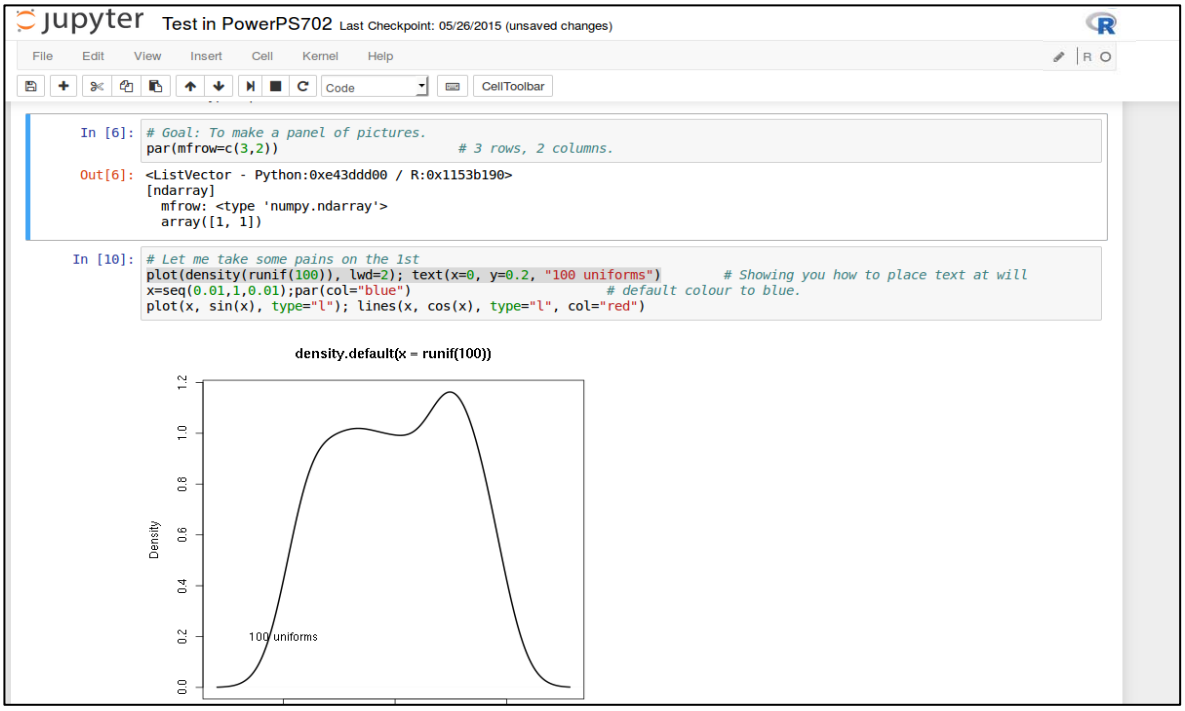

Figure 7 – Screenshot of a Jupyter notebook

The application is interactive, and each notebook is divided into cells where the user can provide the corresponding input (In[] section of the cells) including R statements, as well as markdown input. When the user process the different input cells (one by one or the entire notebook) the outcome appears in the corresponding cells (Out[] section of the cells).

One of the key practical questions is how to provide the access to the final user to these interactive applications, and preserving also the rights to access the data to be processed and the corresponding output, as well as the notebook content. The natural solution is to provide an AAI mechanism in front of them, and link it with the corresponding Jupyter server, typically running in a given VM in the Cloud. A solution being explored is based on the Jupyter hub project[10]: a server that gives multiple users access to Jupyter notebooks, running an independent Jupyter notebook server for each user.

Another important question is how to benchmark the resources in the e-infrastructure regarding the use of R. Preliminary work is presented in the Appendix.

[10] See https://github.com/jupyter/jupyterhub

4 Examples of current services using R in LifeWatch

In this section we describe some of the services using R currently deployed in the LifeWatch community, to motivate the idea of a common architecture to be implemented in FedCloud. Such a common architecture is presented in section 5. The services in this section are described using relevant elements of the EGI 'Service Design and Transition Package'[11]. This will simplify the task of including these services – if appropriate – in the EGI Service Portfolio.

Tools/services at IFCA

Tool name	*TheRmocline*
Tool url	*https://github.com/ferag/theRmocline*
Tool wiki page	*https://github.com/ferag/theRmocline/wiki*
Description	*This script calculates the parameters needed to define the theoretical thermocline using a set of data that describes a vertical profile of the water temperature. Parameters are defined in the wiki page.* *To calculate some parameters, the Script need n R function for optimization that solves the equation for a given formula, in this case a function. The input needed is one or more sets of water column profiles of water temperature.*
Customer of the tool	*LifeWatch team*
User of the service	*This tool has different stakeholders. On one hand, the script can be used by biologists or environmental scientists to study the evolution of the thermocline along time, comparing charts, checking the slopes etc. On the other hand, these parameters are used to validate profiles generated by models like those implemented in DELFT-3D.*
User Documentation	*See above*
Technical Documentation	*See above*
Product team	*IFCA, Advanced Computing and e-Science group*
License	*MIT license*
Source code	*https://github.com/ferag/theRmocline*

Tool name	*Other Scripts: NoDataAlert, LevelPlot*
Tool url	*https://github.com/ferag/Rscripts*
Tool wiki page	*https://github.com/ferag/Rscripts/wiki*
Description	*NoDataAlert scripts queries a database to determine if there was any problem at the time of inserting data. This helps scientists to find out different basic problems in data collection.* *LevelPlot creates a special type of chart of water column profiles with different parameters that can be used for exploring the status of the*

[11] https://documents.egi.eu/document/2550

	water column along a given data collection period.
Customer of the tool	*LifeWatch team*
User of the service	*This set of scripts are used by the automatic processing chain on RAW collected data towards the PROCESSED data.*
User Documentation	*See above*
Technical Documentation	*See above*
Product team	*IFCA, Advanced Computing and e-Science group*
License	*MIT license*
Source code	*https://github.com/ferag/Rscripts*

Tools/services at VLIZ

Tool name	*Lifewatch data explorer*
Tool url	*http://rshiny.lifewatch.be/ and http://rstudio.lifewatch.be/*
Tool wiki page	*See above*
Description	*This interactive tool gives access to all data collected in the scope of the Flemish LW project.* *It is based on RShiny server, leaflet, ggplot, plotly, dynagraph, datatables.* *Some data in moratorium period is protected by password access.* *The system can query MSSQL, PostgresQL, Geoserver (WFS) and MongoDB servers.* *In a first step the serve is queried using basic selections: time period, sampling frequency, project, tag codes etc. This generates an R data frame.* *In a second step the user can calculate additional columns like tidal level, moon illumination, solar cycle, year, month etc.* *In a third step the user can select columns to filter on.* *The user can then download the data, or visualise it in map or time series, scatter plots, box plots etc.* *Several options for adjusting the output interactively are offered.* *If the number data points is too high, the system will either dimish the resolution, or fall back to less interactive plots.*
Customer of the tool	*Scientific public and project partners.*
User of the service	*The Rshiny tool is open to the scientific public; we intend to use it as the main interface for all data collected in the project.* *We have foreseen a password protected access to data under moratorium for the scientists that participate in the project.* *The Rstudio server is for now restricted to the scientists that participate to the project. The data access API used for the Rshiny tool allows them to access the data directly, giving more possibilities for analysis. They can also upload additional data to analyse.*
User Documentation	*In progress*
Technical Documentation	*In progess*

Product team	*VLIZ*
License	*In progress*
Source code	*In progress*

Tools/services at HCMR

Tool name	***Rvlab***
Tool url	*https://rvlab.portal.lifewatchgreece.eu/*
Tool wiki page	*See above*
Description	***R Statistical Processing vLab*** *Towards the alpha version of the RvLab* *The activities that have been carried out towards implementing the first (alpha) version of the RvLab can be broken down into the following sub-tasks:* *1. Determination of the desirable set of functions that the vLab will support.* *2. Analysis of the requirements that these functions introduce to the underlying infrastructure, in terms of computational effort, storage capacity etc. In parallel, an analysis of the features that characterize the datasets to be used as input to these functions has also been made.* *3. Analysis of the available tools and methodologies that can be used, in order to meet the requirements identified in the previous step, for each function.* *4. Development of a generic methodology to be followed for the optimization of the functions.* *5. Application and adaptation of the methodology developed in the previous step to specific functions and generation of different variants.* *6. Configuration and parameterization of the underlying infrastructure (cluster, database etc), to support the execution of the RvLab operations.* *7. Development of an intuitive User Interface to enhance the interaction of the user with the RvLab and present the outcome of the statistical analysis operations.* *8. Integration of the UI with the underlying infrastructure, as well as with the general LifeWatch infrastructure that is under development.* *Below we present more detailed information about the progress achieved within each step.* ***1. RvLab Statistical Analysis Functions*** *RvLab has concentrated its focus on the majority of the functions supported by the vegan CRAN package, which provides methodologies for the analysis of ecological communities. It has tools for analyzing ecological diversity, and for the multivariate analysis of communities (NMDS, pCCA, pRDA etc.), such as diversity analysis, species abundance models, analysis of species richness, ordination, support functions for ordination (dissimilarity indices, extended dissimilarities,*

	Procrustes analysis, ordination diagnostics, permutation tests), dissimilarity analyses (ANOVA using dissimilarities, ANOSIM, MRPP, BIOENV, Mantel and partial Mantel tests) and others. *Moreover, RvLab has extended its support to the optimization of primitive operations that are commonly used by the aforementioned vegan functions.* *RvLab aims to improve vegan functions for diversity analysis, ordination and analysis of dissimilarities, through parallelization and optimization. In order to fulfill this task Rvlab developers have employed modifications on the functions listed below:* *-taxa2dist: finds indices of taxonomic diversity and distinctness, which are averaged taxonomic distances among species or individuals in the community.* *-taxondive: finds indices of taxonomic diversity and distinctness, which are averaged taxonomic distances among species or individuals in the community.* *-metaMDS: performs Nonmetric Multidimensional Scaling (NMDS), and tries to find a stable solution using several random starts. In addition, itstandardizes the scaling in the result, so that the configurationsare easier to interpret, and adds species scores to the site ordination.* *-vegdist: computes dissimilarity indices that are useful for or popular with community ecologists.* *-anosim: Analysis of similarities (ANOSIM) provides a way to test statistically whether there is significant difference between two or more groups of sampling units.* *-adonis: analysis of ecological community data (samples X species matrices) or genetic analysis of ecological community data (samples X species matrices) or genetic data where we might have a limited number of samples of individuals and thousands or millions of columns of gene expression data.* *-mantel: finds the Mantel statistic as a matrix correlation between two dissimilarity matrices.* *-radfi: construct rank - abundance or dominance / diversity or Whittaker plots and fit brokenstick, pre-emption, log-Normal, Zipf and Zipf-Mandelbrot models of species abundance.* *-bioenv: finds the best subset of environmental variables, so that the Euclidean distances of scaled environmental variable have the maximum (rank) correlation with community dissimilarities.* *-hclust: Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing it.* *-simper: Discriminating species between two groups using Bray-Curtis dissimilarities.* *-princomp: performs a principal components analysis.* *2. Requirements Analysis and profiling*

	An extensive requirements analysis and profiling has been carried out on the functions of the vegan package, in order to determine their demands in terms of computational effort, memory usage during execution (primary or secondary) and encoding methodology. The primary objective has been to identify opportunities for optimization of their performance, as well as for offering alternative behaviors (e.g., combination of functions). It is important to note that not all functions can be optimized for parallel execution over the LifeWatch cluster, due to restrictions related to their language of encoding. *An analysis that has been conducted, before implement our methodology, aimed to inspect the aforementioned functions from various perspectives (reason of computational effort, storage usage, function's frequency and internal and external interconnection etc). R's profiling tools, such as profr, proftools, grid, Rgrapviz packages, has supported this analysis.* *After reviewing the results, we identified candidate functions and alternatives for optimization, as follows:* *- taxa2dist: this function is intensive both in terms of computation and in terms of data usage. We parallelized certain of its internal functions, resulting in significant gains in performance. We further offered the ability for alternative data storage, in order for operations that cannot be executed on main memory to use the Postgresql database.* *- taxondive: similarly to taxa2dist, optimization using parallel versions of the underlying operations has been carried out.* *- Taxa2dist + taxondive: our analysis in collaboration with the Lifewatch researchers revealed that these two functions are typically executed in sequence. Since they both share similar manipulation of the underlying data, an integrated version of these functions has been implemented, significantly improving their performance* *-vegdist: despite its popularity in vegan operations, no improvement can be carried out, as the encoding of this function is in C* *-anosim: The parallel version of anosim that has been developed showed that the efficiency of parallelization can optimally exploit the capacity of the LifeWatch cluster.* *-mantel: Similarly to anosim, the parallel version of mantel has achieved significant performance gains.* *Similar analysis is also being conducted for the other functions of the vegan package.* *3. Tools for Optimization of Execution* *The analysis conducted in the previous step revealed ample*

	possibilities for improving the performance of certain computationally intensive vegan functions. It also pointed out the need to exploit alternative means of storage, since in certain cases the bi-products during computation are so demanding in memory resources that no ordinary computer can support. *With respect to parallelization, Rvlab developers investigated the features of popular packages, such as snow, multicore and parallel. Due to the complexity of tasks required in vegan, as well as in other packages for the Lifewatch project, this type of approach for parallel computing was found to be rather restrictive. Hence, Rvlab employs a low-level approach and implements custom parallel solutions which allowed for greater flexibility during optimization protocols. Towards this goal, packages that provide interfaces to MPI for R were utilized. Examples of these include* ***Rmpi****, which permits import low level MPI functions into R, abstracting the complexities of writing C or Fortran code. The more recent* ***pbdR*** *package also offers such a wrapper though the pbdMPI library, which is intended for Single Program Multiple Data (SPMD) programming with Big Data.* *After considering the benefits offered, Rvlab adopts pbdMPI as the primary package for parallelization within LifewatchGreece and is also coupled with other solutions for parallelization or optimization of code, where necessary.* *Regarding the handling of memory, a similar approach was adopted and popular packages were investigated. Working with large datasets in R can be cumbersome because of the need to keep objects in physical memory. The need to keep whole objects in memory creates challenges to those who might want to work interactively with large datasets. Several packages attempted to overcome problems with accessing big volumes of data, like R.huge, ff, filehash, yet their performance were not satisfactory for large-scale projects.* *The bigmem and pbdDMAT packages were also assessed, but were not flexible in handling complex constructs, because of the fact that they relied on their own constructs to handle big data.* *Given the conclusions reached from the aforementioned analysis of tools and packages, the methodologies described next was designed. This aimed at combining the facilities offered by an external database, namely PostgreSQL, and the use of R packages, such as dplyr and RPostgreSQL to connect the R scripts with the database, in order to store and retrieve the necessary data.* *4. Generic Methodology*

The main contribution of the activities conducted so far was the development of a generic methodology that combines experience obtained with the aforementioned packages. The challenge was to combine in a harmonious manner the solutions on the parallelization level with those on the database storage, and not just to integrate them monolithically. Moreover, the methodology aimed at being flexible enough to be adapted to the different requirements of each function.

5. Adaptation of the methodology

The general methodology described above is meant to work as a general rule of thumb for demanding tasks, but is not applied as is in all cases. Efforts in the last period have focused on how best to adapt it for the requirements of each particular function individually. For instance, while taxa2dist can take advantage of both parallelization and secondary storage facilities, anosim only demands the former.

Furthermore, we also combine diverse functions that are commonly used together, in order to end up with a workflow of analysis that is optimized for the infrastructure on which RvLab is installed. An example is the sequential execution of the taxa2dist and taxondive functions.

A set of different variants of vegan functions have been produced, which is to be expanded with more operations in the months to come. The assessment of their performance is another task running in this period, which will become more intense in the last phases of the project.

6. RvLab interface

R's environment offers a convenient way to construct RvLab interface through an online web application that allows for a user friendly graphical interface for efficiency and ease execution of Rvlab functions. The online application was implemented by combining a series of web development languages like HTML and PHP thus making a web interface that is directly linked to the PC cluster at HCMR. The alpha version of the Rvlab is available here http://rvlab.portal.lifewatchgreece.eu/. RvLab is available to users after login to the system. On the main page users can find links to a comprehensive and self-explainable tutorial on how to operate the basic functions of the Rvlab and also how to navigate around the web application.

Customer of the tool	*Currently RvLab is available for individual researchers with human to computer interaction, or computer to computer interaction. Services are also accessed by mobile/tablet application. In the near future we would like to connect with several research providers such as environmental data providers and we want to be able to serve LifeWatch infrastructure or any other infrastructures involved with EGI*
User of the service	*Scientific Community, Researchers, Academicians, Students*
User Documentation	*https://rvlab.portal.lifewatchgreece.eu/files/RvLab_manual.pdf (Available, after login)*
Technical Documentation	*https://rvlab.portal.lifewatchgreece.eu/help/technical_documentation (Available, after login)*
Product team	*HCMR, FORTH*
License	*MIT license*
Source code	*Available upon request*

5 Service architecture

In this section we make an attempt to define a common architecture to offer R execution services. The architecture is based on EGI HTC/HPC resources to provide a scalable execution platform for large communities and large use cases. The service architecture provides an overview of the key (logical) service components and their dependencies to help better understand the structure and logical as well as technical setup of the service.

High-Level Service architecture oriented to the Cloud framework

The following schema shows the proposed architecture defined to include R services within EGI Federated Cloud and link them with external resources if needed:

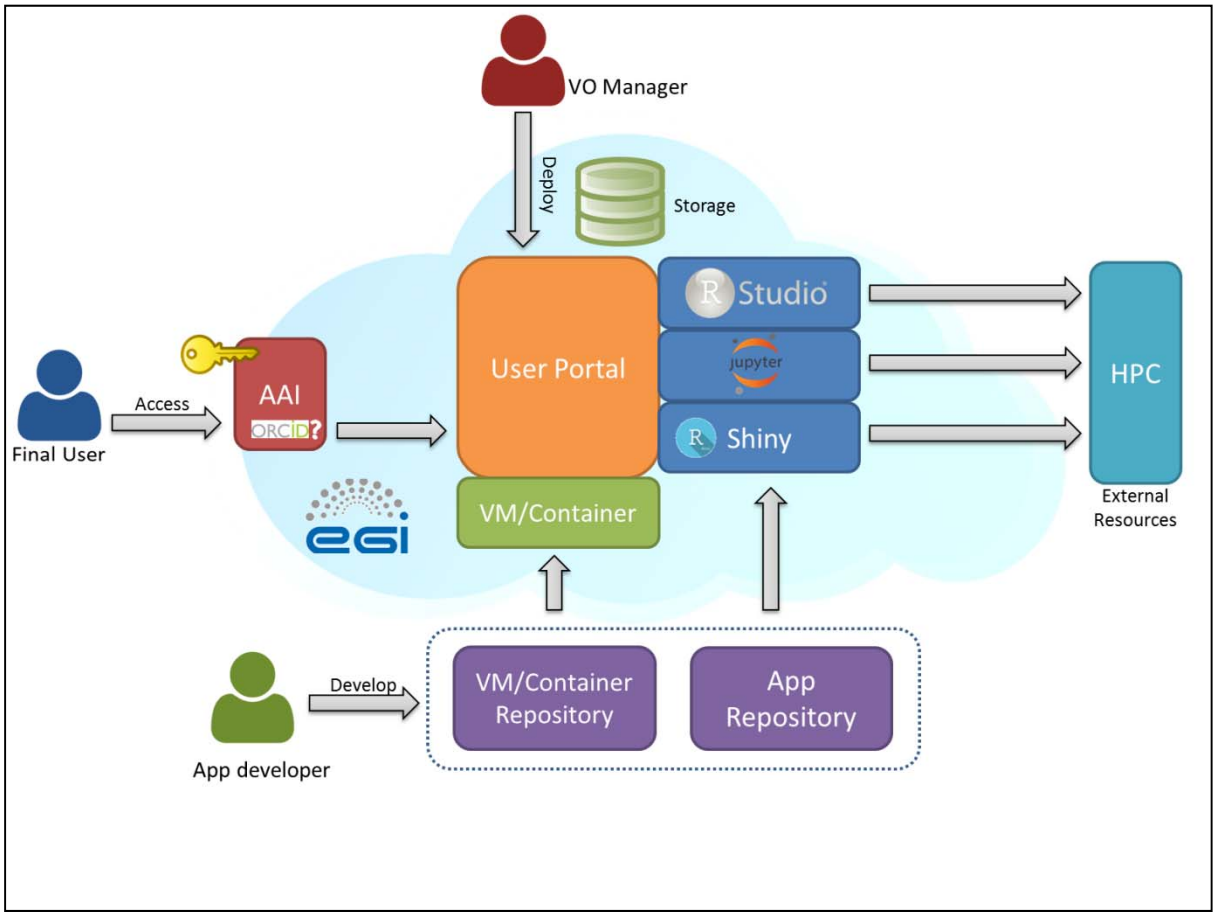

Figure 8 – Service Architecture oriented to the EGI Cloud

The proposed architecture includes three different roles for interacting with the system: VO Manager, App developer and Final User. The Virtual Organization Manager is the person who takes care of the different managerial tasks from the research community

side, in this case, of the management of the Lifewatch Virtual Organization (vo.lifewatch.eu). This person authorizes new users in the organization, that will be allowed to access to the services offered by the infrastructure, and he/she is also the responsible of deploying and administrating general and long term services, like the LifeWatch project management system, GIS servers or R services. The second role, App developer, groups all the IT members who are in charge of developing and eventually deploying new services in the LifeWatch environment such as Virtual Machines or Containers with pre-defined web services, applications based on tools like R shiny, workflows, scripts for Jupyter, etc. The last role, Final User, corresponds to researchers (including biologists but also other with different technical profiles) that consume the resources provided by the infrastructure accessing to the web services: R Shiny final apps, RStudio server, Jupyter notebooks, and also other related tools like GIS or predefined python workflows, etc. The Final User role could also include citizen scientists with access to certain services.

Regarding the AAI (Authorization and Authentication Infrastructures) component, we need a standardized solution to allow different type of final users to access to the infrastructure. Within EGI infrastructure, some solutions are being developed and LifeWatch requirements have been communicated through the specific AARC project[12], where EGI is one of the partners. For researchers, one of the possible solutions is accessing through ORCID credentials, linked to their host institutions. For citizen scientist, an OpenID or similar solution could work. Once the user is logged in, he/she would be able to access to the web services, using local storage or remote cloud storage, and access to collaboration data, including open data, via federated resources, as being explored in the Data Commons solutions[13]. R services are very oriented to data analysis, so that an optimal access to data is key, and it should be closer enough to the computing. That is why users need a storage space accessible by the R services that can be owned by them or by the group that they belong to.

A key component of the architecture is the User Web Portal where the services are published. These services are deployed at sites that provide resources to EGI FedCloud supporting LifeWatch VO. Different types of R services (one or more in each portal) can be published: RStudio server, Rshiny or Jupyter, all of them previously described in section 3. This web portal is implemented in a Virtual Machine (or Container) that can also run over EGI FedCloud infrastructure.

[12] Authentication and Authorisation for Research Collaboration, AARC, see https://aarc-project.eu/
[13] https://wiki.egi.eu/wiki/EGI-Engage:WP4#TASK_JRA2.1_Federated_Open_Data

The services could also be extended to access external HTC/HPC resources (accessed either through 'grid' or 'cloud' interfaces), using specific packages, as explained later. Permanent and general use web portals are deployed by the VO Manager, while temporal services can be deployed by app developers or by final users with LifeWatch VO credentials.

App developers will mainly handle two types of resources: Virtual Machines and Containers predefined to be launched OR Applications based on other services, like Rshiny. Virtual Machines or containers are a set of packaged services ready to be launched in the EGI FedCloud infrastructure and that can be managed via a repository like EGI AppDB[14]. The development of applications based on services like R shiny, what is very focused on providing R access using web interfaces, can also require a *git* repository that could be deployed directly in connection with the corresponding web portals.

Finally, certain R services could require not only local resources but also external HPC resources. In such cases R must be installed in the external resource with specific libraries enabling remote R computing, as explained later.

Challenges

There are some elements of the architecture that need to be integrated with others to be completely functional: AAI, access to external resources, access to storage and access to repositories.

[14] At the time of writing AppDB already supports VMs, but does not (yet) supports containers. Containers can be manually deployed into VMs, for example as described at https://wiki.egi.eu/wiki/Federated_Cloud_user_support#Docker_containers.

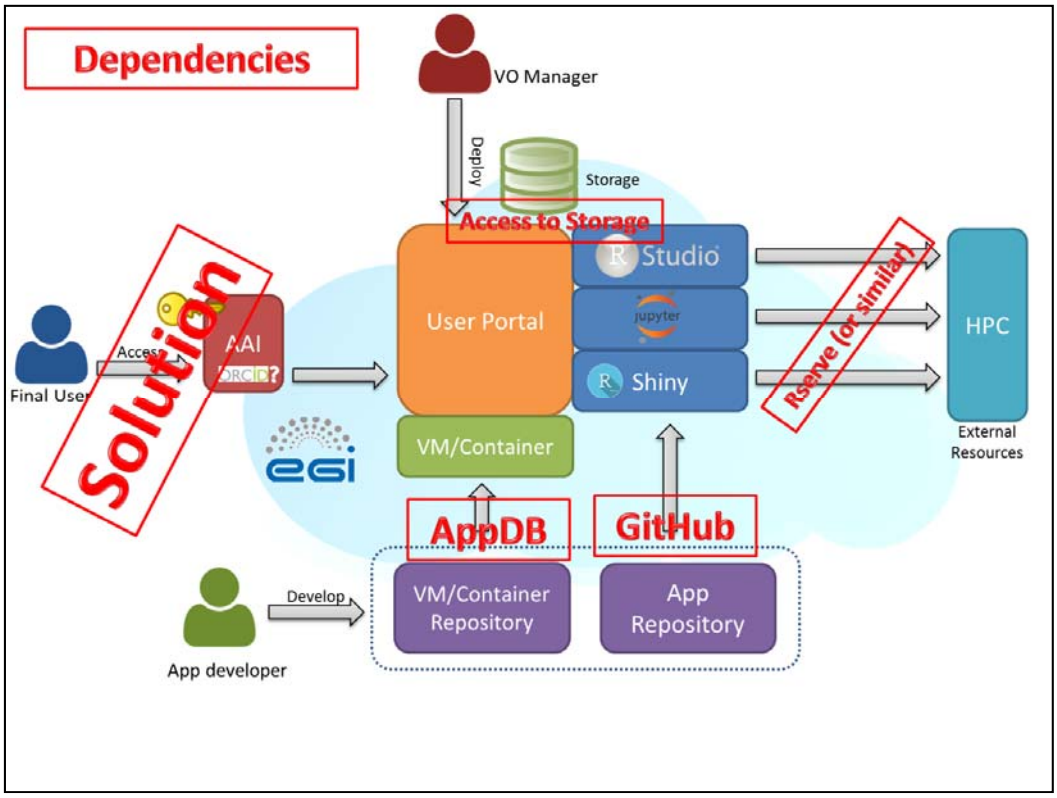

Figure 9 – Service Architecture: Dependencies

AAI

For AAI, as previously indicated, EGI is working on providing a general solution for certain type of users, like a SSO (single sign-on) solution for accessing services. This development is ongoing in the the JRA1.1 task of the EGI-Engage project[15]. This work aims at a pilot system that would

1. Simplify the process of connecting EGI services (e.g. AppDB, Operations Portal, GOCDB, etc.) with AAI architectures operated by external infrastructures, such as LifeWatch.

AND

2. Harmonise the integration of EGI services across multiple, externally operated RI AAIs. (e.g. AppDB would be connected to the LifeWatch AAI, the DARIAH AAI, the ELIXIR AAI in a harmonised way).

The design of this new EGI AAI pilot system has finished in 2015 in close collaboration with the AARC H2020 project . In the heart of the pilot system there is an 'IdP/SP Proxy' component, which is based on SAML technology. The IdP/SP Proxy will be responsible for

[15] https://wiki.egi.eu/wiki/EGI-Engage:WP3#TASK_JRA1.1_Authentication_and_Authorisation_Infrastructure

mapping an external user identity to an 'EGI identifier' which will be used for the same user across all the EGI services. The IdP/SP Proxy will be able to import attributes from external attribute authorities (e.g. from LifeWatch IdPs) and assign these to the internal EGI user identifier. Based on the imported attributes the EGI services can authorise users across the whole EGI network in a coherent way.

Another relavant development in EGI is the recently established 'Long-tail of science platform'[16], and particularly its user registration and authorization system. In this platform a 'User Registration Portal' serves for users to request access to the infrastructure. After a user request is approved, the allocated capacity can be accessed trough any of the scientific gateways (VREs) that are connected to the platform. The gateways use robot certificates with a special extension to separate users, and to allow complete tracking of user activities at the lower levels of the infrastructure.

The R community is pretty active on developing new packages that enrich the features of a basic R installation. As indicated before, the Rserve package enables consuming resources from an external R installation, i.e. using external computational capacities. To do so, R must be installed in both resources (client and server) as well as Rserve packages: the client needs to query the server and the server must be able to get that request, compute it and return a result. This system can be used to consume external HPC resources. There are other interesting solutions like the option provided in RStudio Server Pro, which is able to manage load balancing using different computing nodes, but unfortunately this is not available as an open source solution yet.

Access to data

Regarding the critical access to data storage, we could use a distributed storage solution, like LUSTRE or GPFS, deploying a global file system. In such case, every R server needs to be able to access to that global file system, and it needs to be flexible enough to accept new clients, or mount the corresponding volume with the well known security problems. As indicated before, within this EGI-Engage project and under the development of Data Common, EGI is testing a general solution for distributed storage based on OneData (https://onedata.org/). This solution is a very good option to be used within the LifeWatch framework.

Repositories

Finally, the last dependencies to be considered are related to repositories. Virtual Machines or Containers can be stored in AppDB by EGI, so the link between the repository

[16] https://wiki.egi.eu/wiki/Long-tail_of_science

and the infrastructure is direct. However, a customized deploying system needs to be developed to work with application repositories, like GitHub. This deployment can be addressed at the time of setting up the VM/container or later on, after launching it.

6 Feedback on satisfaction

In this section we report briefly on the positive, although yet limited, experience with the use of the R tools described before. Part of the work presented here has been done under different initiatives closely related to LifeWatch.

Experience at IFCA

The option explored to run the R scripts by the final user (D.G. working for Ecohydros SL) was the use of Jupyter notebooks. As indicated before, notebooks are organized in different cells, so users can work in different parts separately and wrote the script in different parts and then check one by one if there is any error. This option was considered quite useful by D.G.

As an example, when executing the *theRmocline.R* script, all necessary parameters are first checked and then the fit of the simulated thermocline versus the actual data is done (see below).

```
        #Parámetros derivados
        Zt <- (1/alfa)*((1-(1/n))^(1-(1-(1/n))))
        TZt <- Th+((Te-Th)/((1+((alfa*(Zt))^n))^(1-1/n)))
        TZtprima <- (-1)*(((Te-Th)*(n-1)*((alfa*Zt)^n)*((((alfa*Zt)^n)+1)^((1/n)-2)))/(Zt))
        Zu <- Zt-((TZt-Te)/(TZtprima))
        Zb <- Zt-((TZt-Th)/(TZtprima))
        S <- (Te-Th)/(Zb-Zu)

        #Muestro y guardo la gráfica
        mod <- data.frame(x=mod.RSS(dat, c(alfa,n)),y=dat$y)
        plot(y ~ x, data = dat)
        title(paste("Perfil: ",file))
        lines(mod$x,mod$y)
        if (!is.nan(Te) && !is.nan(Th) && !is.nan(alfa) && !is.nan(n) && !is.nan(Zt) && !is.nan(TZt) && !   is.nan(TZtprima
        {
            insert = paste('Te =', round(Te,2),', Th =', round(Th,2), ', alfa =', round(alfa,2), ', n =', round(n,2), ', Z
            print(insert)
        }
    }
    return (c(Te, Th, alfa, n, Zt, TZt, TZtprima, Zu, Zb, S))
}

In [21]: paramTermoclina("OceanDischarge01_09_14.xyz")

[1] 10.55905
[1] "Te = 21.59 , Th = 10.56 , alfa = 0.07 , n = 5.26 , Zt = 13.95 , TZt = 17.38 , TZtprima = -0.93 , Zu = 9.43 , Zb = 21
.27 , S = 0.93"

Out[21]:  21.589844  10.559053  0.0688774216120351  5.25801354952523  13.9475941786974  17.3819848481672  -0.932030530469676
          9.43287173745678  21.268096699928  0.932030530469676

In [ ]:
```

Figure 10 – Execution of an R script in a Jupyter notebook

The script has two versions: one reading data from CSV files and another one using directly the database, via the RMySQL package. If this script is used only to fit a single thermocline, there is not much difference versus using a workstation at the office. However, as many files corresponding to the output of different models were to be compared with the data taken each day along the hydrological year (around 300 vertical profiles), the computation time was significantly reduced.

The main advantage to work with Jupyter notebooks and R instead of using Excel (the usual solution in the company for data analysis) is the option to automate the process of uploading many different files and calculating all the fit parameters in one round. Another important

advantage is that the script exports the results from the study and the charts and automatically saves them in pdf format, completing the whole process.

Experience at VLIZ

The Lifewatch data explorer tool has been already tested by several project partners, and their feedback has largely been implemented. A training workshop will be organized in the next couple of months.

The system suffers from performance issues when dealing with very large datasets, and some measures have been taken to compensate for this. The main problem is on the client side (the browser), so we moved more of the processing load to the server where R is able to handle higher demands. How the servers 'scale' when more users get involved is uncertain for now. Mirroring, clustering, load balancing the servers is possible, but not a priority at this stage.

Experience at HCMR

Rvlab has been tested in numerous workshops and training courses including the EMBOS workshop in Crete, 2014 (https://www.lifewatchgreece.eu/?q=content/embos-synthesis-meeting-0), the Lifewatch data analysis workshop (biodiversity data preparation and analysis using the Lifewatch virtual labs and web services) in VLIZ, 2015 (http://lifewatch.be/en/lifewatch-data-analysis-workshop) as well as other meetings and conferences.

7 Future plans

The described services are currently deployed both in local resources (that can be linked to EGI FedCloud environment) and in external sites that support Lifewatch Virtual Organization. In the last months a new site integrated with EGI FedCloud has been deployed in "Estación Biológica de Doñana", placed at Seville, Spain. This site will serve as the main site providing resources to the Lifewatch Virtual Organization, including both central services and user-oriented services, like Geographical Information Systems, Databases, Data Catalogues or computing-analysis services based in different technologies like R. That is why in the near future the distributed services that are very important for Lifewatch users will be installed at EBD site, including those explained in the previous sections.

The features provided and the infrastructure deployed at Seville, will increase the interest on joining the LifeWatch Virtual Organization. For instance, Seville site has deployed a large NFS system that can store data from users and connect to the computing part based on Cloud in a faster way. Also, this NFS system can be integrated with the OneData Solution that is being tested by EGI as a distributed storage solution.

In parallel, in the framework of the LifeWatch EGI-CC, several working groups have been established in the past meeting at Bari EGI Conf 2015, and one of them lead by HCMR and with the collaboration of VLIZ and IFCA teams, will address the support to R solutions. This work is now being tracked using the OpenProject web tool, that has been selected by LifeWatch as the basic support tool for its distributed e-infrastructure project, and in this way we expect to collect in the next months the requirements of more Case Studies, analyze them and create the corresponding backlog focused on R services.

Also the work on the integration of an EGI AAI solution has already started, in connection to the AARC initiative.

Finally, and as clearly indicated in the example of VLIZ marine observatory, a global framework for the development and use of the different tools and services is already implemented: the LifeWatch Marine Virtual Research Environment (LW Marine VRE, see http://marine.lifewatch.eu/). The LifeWatch EGI-CC will support this development as well as the ongoing similar effort on other areas (Non-marine VRE).

Appendix: Testing R in HTC and HPC resources

R can be installed in many different infrastructures, including HPC systems, HTC clusters and Cloud services. Depending on the use case or the problem to solve, choosing the right infrastructure can be significant in terms of execution time. For instance, HPC systems could be the best option for parallelized scripts that have to be executed, using a package implementing that parallelization like Rmpi. Furthermore, the difference in the processors used by the "physical" machines can also reflect in the execution time, so this short landscape includes also the experience comparing also Intel-Xeon and IBM-Power processors.

R in an HPC system: Altamira supercomputer

Altamira is one of the nodes of Spanish Supercomputing Network, RES. Currently, Altamira comprises 158 main compute nodes, 5 additional GPU compute nodes and several service servers. All main compute nodes have two Intel Sandybridge E5-2670 processors, each one with 8 cores operating at 2.6 GHz, 64 GB of RAM memory (i.e. 4 GB/core) and 500 GB local disk. The system also includes five nodes with GPUs and eleven Power PC servers.

R is used in Altamira for different purposes: economics, statistics analysis, meteo applications, biomedicine, environmental analysis, etc. R packages sometimes are limited to be used with a certain R version and cannot be used with another, but Altamira uses a modular system for software that allows the users to select and load the R version that they need to run a package. Furthermore, R allows users to select a customized workspace not only for input and output files but also for package installation, so users in Altamira do not need to be administrators to download and use new packages.

The actual command to load R before using it is given below:

```
[userX@login1]$ module load R
    load R/2.15.1 (PATH,MANPATH)
```

R Parallelization

Different tests have been performed in Altamira in order to know the performance of the installed R packages that can benefit of parallelization using a different number of cores. In particular, a matrix multiplication test has been executed using 1, 2, 4, 8 and 16 cores. The results are shown in the figure below: a reasonable speedup is found relevant for processes taking a relevant execution time.

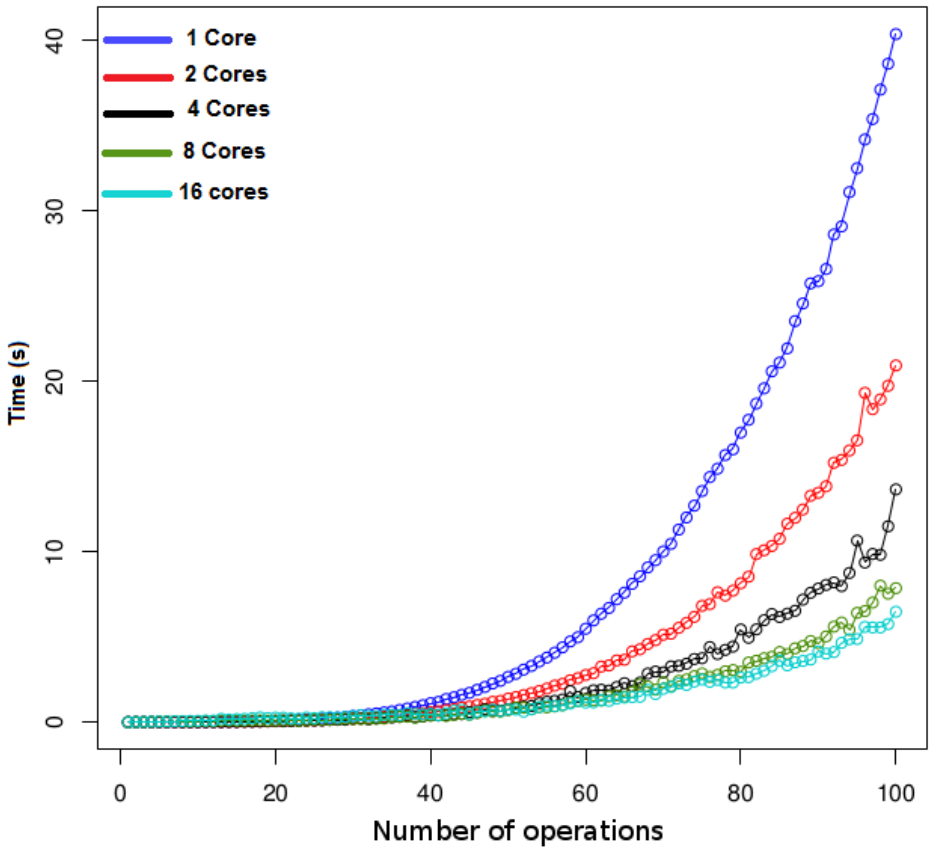

Figure 11 – Matrix Multiplication using R parallel options in a node in Altamira

Installation of R in other HPC systems

Altamira system also includes a cluster with IBM POWER7 blades with a capacity for up to 700 processes to execute intensive CPU applications. POWER7 processors are well suited for computing intensive applications, and R was installed in these systems, and we share here the details as a path to installation on other non-Intel processors based systems.

Before installing R, the system must have a compiler available for the powerpc architecture installed: ppc64. If not, GCC must be installed as the basic compiler that integrates other compilers like gfortran and g++. Besides, GCC is optimized by IBM in a package, IBM Advance Toolchain for Power Linux. The Toolchain version installed is 8.0 which gives support for big endian (ppc64) and little endian (ppc64le) POWER7 and POWER8. So, installation requires the following steps:

1. Configuring the repositories adding the line to **/etc/apt/sources.list**:

deb ftp://ftp.unicamp.br/pub/linuxpatch/toolchain/at/ubuntu trusty/ at8.0

2. After configuring the repository, update apt:

$ sudo apt-get update && sudo apt-get upgrade

3. Finally, install the package Advance Toolchain:

$ sudo apt-get install advance-toolchain-at8.0-runtime \ advance-toolchain-at8.0-devel \ advance-toolchain-at8.0-perf \ advance-toolchain-at8.0-mcore-libs

4. After complete installation, the tool is ready to use. For example, gcc with the command:

$ /opt/at8.0/bin/gcc

The next step is to install the appropiate version of R from source, R-3.1 3, and compile it. We chose that version because it is supported by the features of the system and for installing the needed libraries and packages required by the R services explained in next sections.

After installation, users can run R interactively typing the **R** command in the terminal.

Comparing performance: R Benchmarking

R-Benchmark 25 is a set of benchmarks developed by the R community that allows testing different complex operations within the R environment in order to check the performance.

The benchmark code is available in http://www.revolutionanalytics.com/revolution-revor-enterprise-benchmark-details.

Using this benchmark we have compared the performance of different "typical" systems, using a default R installation[17], as described before. The list of systems include:

-LAPTOP (using an i5 processor)

-WORKSTATION (using a Xeon E3 processor)

-ALTAMIRA node (using a Xeon E5 processor)

-PS701 (using a Power7 processors)

-CLOUD (running a VM on top of a Xeon E5 processor)

In the next table we present the benchmark results (in seconds, less is better) for the systems tested. There is no use of parallelization, so the expected key parameters are the processor architecture and its frequency. Along this line, the results obtained are coherent, but it is interesting to observe the difference between Power and Intel-Xeon processors.

The inclusion of a complete parallel test could help to improve the comparison, and this is the subject of our last comparison.

[17] Notice that further work is required to compare different R implementations, as an example see: http://www.r-bloggers.com/r-r-with-atlas-r-with-openblas-and-revolution-r-open-which-is-fastest/

LAPTOP			
CPU	**CORES**	**R**	**RESULT**
Intel core i5-2450M CPU i386 @2.5GHz	2 cores (4 virtual cores)	3.0.2	42.0 s.
WORKSTATION			
CPU	**CORES**	**R version**	**RESULT**
Intel Xeon CPU x86_64 @2.40GHz	4 cores (8 virtual cores)	3.1.3	34.3 s.
ALTAMIRA node			
CPU	**CORES**	**R**	**RESULT**
Intel Xeon E5-2670 @2.60GHz	16 cores	2.15.1	35.1 s.
PS701 blade			
CPU	**CORES**	**R version**	**RESULT**
POWERPC64 @3.0GHz	8 cores (32 virtual cores)	2.15.1	26.4 s.
CLOUD (Virtual Machine)			
CPU	**CORES**	**R**	**RESULT**
Intel Xeon E5-2670 @2.60GHz	16 cores	3.0.2	37.4 s.

Table 1 – Comparison of the R benchmark on different systems

The next table shows the results running different benchmark parts, that show significant differences pointing to the need to carefully select the right system for large R workloads.

	PS701	WORKSTATION	LAPTOP	CLOUD
Matrix Multiply	49.4	132.5	156.1	121.3
Cholesky Factorization	35.4	81.2	87.8	66.4

Table 2 – Comparison of two components of the R benchmark for different systems (s.)

Comparing now in more detail the parallel and non-parallel versions of these benchmark components for the Power and Intel systems, the differences can be further contrasted.

	PS701 Power7		WORKSTATION Xeon E5	
Benchmark component	Parallel	Not Parallel	Parallel	Not Parallel
Matrix Multiply	11.7	49.4	33.1	132.5
Cholesky Factorization	8.4	35.4	19.9	81.2

Table 3 – Comparison of the parallel and non parallel versions for two components of the R benchmark (s.)

Using R services in an HPC system

Following the previous discussion regarding the good performance of Power based systems for R computing, and in order to provide an interactively and web based interface, a Jupyter Notebook Server have been deployed in the Power7 cluster. As already indicated, Jupyter Notebook is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media, etc.

There are two main components:

1. The Jupyter Notebook web application, for interactive authoring of literate computations, in which explanatory text, mathematics, computations and rich media output may be combined. Input and output are stored in persistent cells that may be edited in-place.
2. Plain text documents, called notebooks, for recording and distributing the results of the rich computations.

Inside the Jupyter Notebook the R interface can be run. For security and accessing reasons, an authentication based on certificates is being deployed and tested and it will allow only certain users or Virtual Organizations to log in (like LifeWatch VO).

To install and configure Jupyter notebook, these steps have been followed:

1. Jupyter requires python >=3.3 or python 2.7, so first, install Python and other dependencies:

   $ sudo apt-get install python3-dev build-essential python3-qt4

   The version 3 for python is required to install R kernel in Jupyter notebook.

2. Install a virtual environment where Jupyter notebook server will be deployed. Virtual environment creates a python environment with its own directories and packages that is isolated for the rest of the environments in the system. Jupyter is an environment oriented for a single-user work, so the installation will be made after opening a session for that user. Then,

install the virtualenv tool via pip3 and create the virtual environment that will be allocated inside the folder named jupyterenv. The last point is activate it to install and configure jupyter.

```
$ pip3 install virtualenv
$ virtualenv /usr/bin/python3 jupyterenv
$ source jupyterenv/bin/activate
```

3. In that point, jupyter will be deployed inside the virtual environment for running a R kernel. A kernel in Jupyter is a program used by the users for writting code in a specific language. For that, we only have to install jupyter via pip3, run the notebook server and check if it is running in the default port, 8888, in localhost.

```
$ pip3 install jupyter
$ jupyter-notebook
```

4. Jupyter is very useful if more users can access to the server to create and share their own notebooks. For this case, Apache2, via Proxy, is used to connect the localhost:8888, where is running the server, to a public IP for the remote access. As we explained in the introduction of this section, only certain users can access to the services, therefore, the appropriate parameters, such as SSLEngine and the certificates, will be defined in the configuration before restarting Apache.

5. *Once the user closes the session, Jupyter turns down.* To keep it running, Jupyter server should be run as a service, using Systemd utility as system administrator. As root user, create a new file **/usr/lib/systemd/system/jupyter-notebook.service** and copy the following contents into it:

```
[Unit]
Description=Jupyter notebook
[Service]
Type=simple
PIDFile=/var/run/jupyter-notebook.pid
ExecStart=/usr/bin/jupyter-notebook --no-browser --profile=myserver
WorkingDirectory=/home/user/notebooks
[Install]
WantedBy=multi-user.target
```

The line, ExecStart=/usr/bin/jupyter notebook --profile=myserver specifies the command to start the IPython notebook server as we do previously. For security, the working directory should be a folder within the home directory of the user.

As the root user, reload all the systemd unit files, enable the jupyter-notebook service so that it starts on boot, and then start the service:

$ systemctl daemon-reload

$ systemctl enable jupyter-notebook

$ systemctl start jupyter-notebook

Check if the ipython-notebook is running:

$ systemctl status jupyter-notebook

```
● jupyter-notebook.service - Jupyter notebook
  Loaded:loaded(/usr/lib/systemd/system/jupyter-notebook.service; enabled)
  Active: active (running) since Wed 2016-02-03 13:09:02 CEST; 1 day 2h
```

6. The last point is to install the R kernel and make R available for Jupyter server. After opening a R session, install the needed packages from the public repostory of IRkernel In Github and the kernel spec for the current user running jupyter. Restart the jupyter service to commit the last changes.

```
>install.packages(c('rzmq','repr','Irkernel','Irdisplay'),repos=c('http://irkernel.github.io/', getOption('repos')))
> Irkernel::installspec()
```

Open a browser and access the IP where Jupyter is running: https://power701.ifca.es . Before the dashboard appears and in order to access the service, introduce your certificate to authenticate your account in the server. Once you view the current files in the dashboard, create a new notebook for R as shown in the figure below.

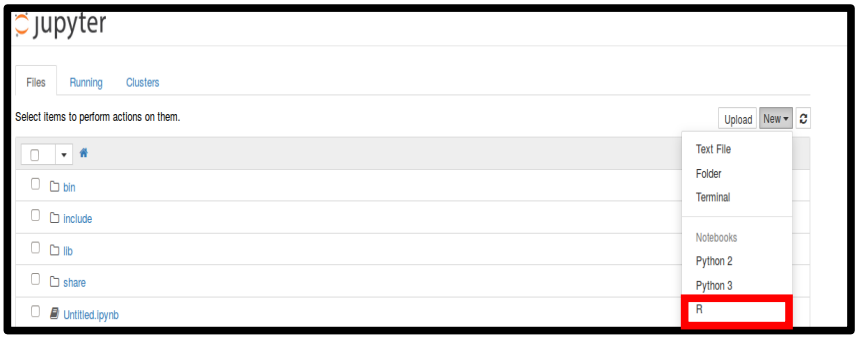

Figure 12 – Screen shot of Jupyter running in ps701 HPC system